Clustering by means of a Boltzmann machine with partial constraint satisfaction

John Spagnuolo, Jr.
James B. Lathrop
California Institute of Technology
Jet Propulsion Laboratory
4800 Oak Grove Drive 301/270
Pasadena, California 91109

Abstract. The clustering problem refers to the partitioning of target sightings into sets. Two sightings are in the same set if and only if they are generated by sensor detections of the same target and are in the same great circle arc (GARC) trajectory of that target. A Boltzmann machine is developed whose sparse architecture provides for only partial constraint satisfaction of the associated cost function. This together with a special graphics interface serve as an aid in determining GARCs. Our approach differs from others in that the neural net is built to operate in conjunction with a non-neural tracker. This further restricts the architectural complexity of the network and facilitates future experimentation regarding decomposition of the neural net across several Von Neumann processors. Also, the Boltzmann machine architecture eases the effort of finding optimal or near optimal solutions. Results are presented. The demonstrated feasibility of neural GARC determination encourages investigation into the extension of its role in the track formation process utilizing an environment that includes supercomputers, neurocomputers, or optical hardware. The network architecture is capable of identifying a host of geometric forms other than GARCs and can thus be used in several domains including space, land, and ocean.

Subject terms: Boltzmann machine; clustering; tracker; parallel annealing schedule.

Optical Engineering 33(1), 251–266 (January 1994).

1 Introduction

The path of an aircraft is typically represented by a sequence of geodesics, great arc circles (GARCs), in the surface of an imaginary sphere surrounding the earth. We are given that n sightings are generated by several sensors and that m GARCs are formed by the various aircraft giving rise to these sightings. The clustering problem is how to decompose these n sightings into m sets such that connecting all the sightings of each set in a time-ordered fashion gives rise to a corresponding set of GARCs, each of which is generated by a targeted aircraft. The problem is nontrivial even for small values of n and m as shown in Fig. 1 (for convenience, GARCs are represented as straight line segments between consecutively detected sightings). Note that single sighting clusters are allowed. Our approach to the clustering problem uses a sparsely connected neural network, which we refer to as a neural clusterer. The design of this neural clusterer allows reasonably fast convergence to an optimal (i.e., globally optimal) or near optimal solution using only a single processor. The inputs to the neural net are the numbers n and m and an $n \times n$ matrix. The i'th, j'th entry of the $n \times n$ matrix consists of a "cost" of associating sighting i in the same GARC with sighting j. These positive costs are generated not by Kalman filter estimates but by algorithms in a companion tracking mechanism that takes into account the velocities and relative positions of the sightings.[1] Associated with each sighting s_x is its location l_x, time of detection t_x, and observed velocity $v_{x,\text{obs}}$. The observed velocity may vary by an amount $\pm\delta_x$. To determine the cost of associating a pair of sightings s_1 and s_2 in the same GARC, it is assumed that the hypothesized single object giving rise to the sightings has traveled from the location of one sighting to the location of the other along a GARC. We assume that the object is traveling from the location of the sighting with the earlier time stamp to the location of the sighting with the later time stamp. For simplicity, let $t_1 < t_2$. The direction of the GARC, the distance between the sightings based on their locations, and the individual time stamps of the sightings give rise to what will be referred to as a predicted velocity $v_{x,\text{pred}}$ for each sighting in the pair. Computations are then made yielding, for each sighting i, a score σ_i that expresses a "match" between the predicted and observed velocities while taking into account the associated δ_i value. Thus, for $i \in \{1, 2\}$,

$$\sigma_i = \Phi(v_{i,\text{pred}}, v_{i,\text{obs}}, \delta_i) \quad . \tag{1}$$

The cost associated with the sightings is then taken simply to be the average of the two scores. Figure 2 illustrates these

Paper 23052 received May 23, 1992; revised manuscript received March 25, 1993; accepted for publication March 26, 1993.

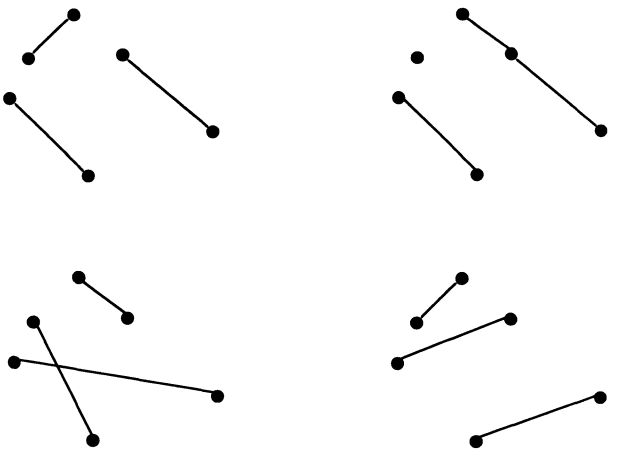


Fig. 1 Possible clusterings for $n=6$ and $m=3$.

ideas in a one-dimensional space. Here, all sightings are in an x-y plane with positions having the same ordinate and velocities restricted to the $\pm x$ direction. As indicated in Fig. 2(a), the observed velocity and error values are $v_{1,\text{obs}}=12.4$, $\delta_1=0.6$, $v_{2,\text{obs}}=-6.4$, $\delta_2=0.3$. Figure 2(b) illustrates an assumed GARC trajectory undertaken by a hypothesized single object giving rise to s_1 and s_2 (thus traveling from l_1 to l_2 in time t_2–t_1). Without loss of generality, it is assumed that the positions and times yield $v_{1,\text{pred}}=12.9$ and $v_{2,\text{pred}}=12.9$, as indicated in Fig. 2(c). The equality of the predicted velocities results from the simple geometry of the situation. In this scenario, more complicated equations symbolized by Eq. (1) would reduce to the form

$$\sigma_i=\left(\frac{v_{i,\text{pred}}-v_{i,\text{obs}}}{\delta_i}\right)^2 . \tag{2}$$

Substituting the appropriate values gives $\sigma_1=0.69444$ and $\sigma_2=4138.7778$, yielding the cost of association between these two sightings as

$$\frac{\sigma_1+\sigma_2}{2}=2069.73612 . \tag{3}$$

This would be a high cost and results from the fact that the observed velocity of sighting s_2 is in a direction opposite to the predicted velocity needed by the object to get from l_1 to l_2 in time t_2–t_1. For higher dimensions, the relevant expressions become more complicated, as indicated in Ref. 1. The cost is always the average of the two scores, however. More recent work develops techniques that give the cost of associating two sightings in the same sequence[2] of two GARCs. This sequence is referred to as a maneuver.

The output of the neural clusterer is a disjoint division into m sets of the n sightings. The sightings in each set are connected to one another in a time-ordered fashion to yield a display similar to that of Fig. 1.

The results of the companion tracker are shown on its own graphics display screen. The user can select various tracks from this display. The number of sightings and the number of GARCs composing the selected tracks together with the associated cost matrix are sent to the neural network. The neural network then yields a grouping of sightings into

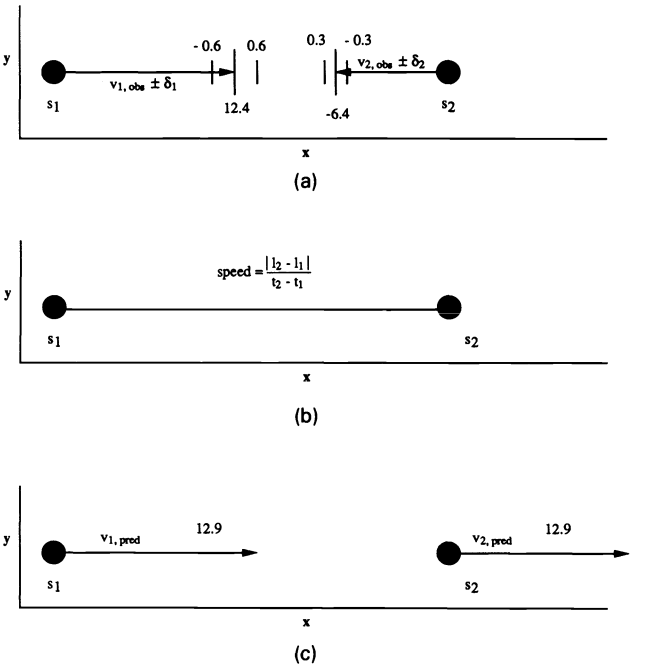


Fig. 2 Simple example for scoring and cost evaluation: (a) observed velocities with errors, (b) assumed trajectory, and (c) predicted velocities.

GARCs using all cost values simultaneously. Because the tracker develops and uses these values iteratively,[3] the neural grouping of GARCs could differ considerably from the tracker's. In most cases encountered thus far, the neural clusterer has agreed with the tracker's results. Both closely mirrored the real-world situation.

There are several studies relating to the use of neural architectures in tracking.[4–7] The approach here differs in that the clusterer is used in conjunction with an already existing tracking system that, as a result of its processing, eases the computational load of the neural net by providing it with a restricted set of parameters. This allows for experimentation with a relatively sparse neural architecture that can serve effectively as a GARC formation mechanism that is amenable to efficient parallel decomposition on Von Neumann architectures. Further, this neural clusterer architecture has the potential for being useful in the actual track formation process now undertaken by the companion tracker. Finally, we use a Boltzmann machine[8] architecture (as opposed to a Hopfield type as used in the mentioned references). This has helped circumvent the problem of getting stuck in local minima.

The neural clusterer is presently running in sequential mode on a Silicon Graphics IRIS 240/VGX. This paper deals more with the correctness of the divisions of sightings into GARCs rather than with the speed of convergence, although efficiency occurred in many examples. The work up to this point demonstrates the feasibility of a parallel decomposition of the GARC formation problem on a Boltzmann machine. Convergence issues can thus be examined in light of the execution of this architecture on neurocomputers.[9,10] Further, because of the increasing feasibility of and momentum toward implementing neural networks in optical hardware,[11,12] and the results of optical implementation of adaptive clustering[13] and tracking[14,15] networks, investigations re-

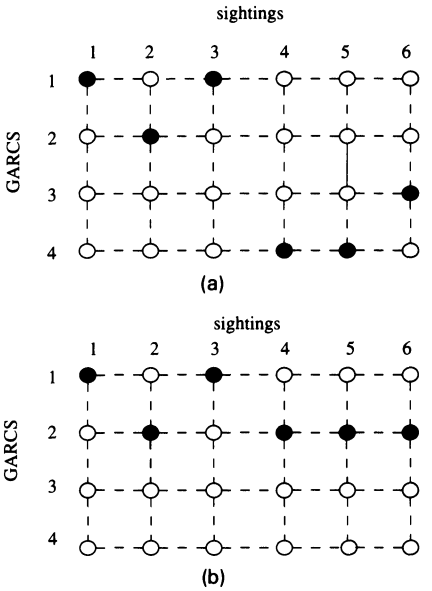


Fig. 3 Types of neural configurations: (a) feasible configuration and (b) semifeasible configuration.

garding the incorporation of the clusterer's Boltzmann architecture into this medium[16] show potential benefit as well.

2 Design

The neural clusterer is a Boltzmann machine. It consist of mn processing elements, where n is the number of sightings and m is the number of GARCs they are to be decomposed into. It can be thought of as an $m \times n$ array of such elements. A neuron that is on (has value 1) in row k' and column n' means that sighting n' is in GARC k'. A disjoint division of the sightings is therefore represented when each column has exactly one processing element with value 1 and each row has at least one processing element with value 1. The nature of the connections as we developed them and the tendency of the neural net to maximize the consensus function ensure that the locally optimal states of the network represent a disjoint division of the sightings among the GARCs. An example of such a feasible configuration where $m=4$ and $n=6$ is indicated in Fig. 3(a).

All empty nodes indicate processing elements that are off (have value 0). From the diagram, we conclude that sightings 1 and 3 are in GARC 1, sighting 2 is in GARC 2, sighting 6 is in GARC 3, and sightings 4 and 5 are in GARC 4.

We denote a processing element in the a'th row and b'th column as e_{ab} and associate it with the variable x_{ab}, which equals 1 if e_{ab} is on or 0 if it is not. Denote by c_{xy} the cost between sightings x and y ($c_{xx}=0$). Let

$$X=\bigcup_{\substack{0\leq s\leq m-1\\ 0\leq t\leq n-1}}\{x_{st}\} \ . \tag{4}$$

To achieve final configurations with this distribution of neuronal patterns, the network is constructed so as to minimize the function

$$f(X)=\sum_{i,j=0}^{m-1}\sum_{\substack{p,q=0\\ p<q}}^{n-1} a_{ijpq}x_{ip}x_{jq} \ , \tag{5}$$

where

$$a_{ijpq}=\begin{cases} c_{pq} & \text{for } i=j \\ 0 & \text{otherwise} \end{cases} \ , \tag{6}$$

subject to the following constraints: First, for any t where $0 \leq t \leq n-1$,

$$\sum_{s=0}^{s=m-1} x_{st}=1 \ , \tag{7}$$

and second,

$$\sum_{s=0}^{s=m-1}\sum_{t=0}^{t=n-1} x_{st}=n \ . \tag{8}$$

Notice that under these constraints, a configuration having one or more 0 rows, as shown in Fig. 3(b), is allowed. Such a configuration is called a semifeasible configuration. It will be shown that a network constructed to minimize $f(X)$ subject to Eqs. (1) and (2) is enough to provide convergence to a feasible solution.

Each processing element in the neural clusterer has a connection going from itself to itself. This is called the bias connection of the processing element. It will be used to make certain that there are no empty columns in the final state of the net. For any two neurons in the same column, there is an inhibitory connection. Its purpose is to guarantee that no more than one sighting appears in each column. Finally, for any two neurons in each row, there exists a connection associated with the cost of allowing the two sightings represented by those neurons to be in the same GARC. We represent the bias, inhibitory, and cost connection sets respectively as C_b, C_{in}, and C_c. If we depict the symmetric connection between processing elements e_{ab} and e_{cd} by $\{e_{ab},e_{cd}\}$, then a more precise description of the neural architecture is given by

$$C_b=\{\{e_{ab},e_{ab}\}\} \ , \tag{9}$$

$$C_{\text{in}}=\{\{e_{ab},e_{cb}\}\} \quad a\neq c \ , \tag{10}$$

$$C_c=\{\{e_{ab},e_{ad}\}\} \quad b\neq d \ , \tag{11}$$

where $0 \leq a,c \leq m-1$ and $0 \leq b,d \leq n-1$. In the next section, we demonstrate how these choices allow for convergence of the neural network to useful solutions.

Basic wiring of the neural clusterer for a net where $n=6$ and $m=4$ is shown in Fig. 4. Vertical and horizontal straight lines indicate connections between adjacent processing elements. The number of connections based on the above scheme is

$$n\left[\frac{m(m-1)}{2}\right]+m\left[\frac{n(n-1)}{2}\right]+nm \ , \tag{12}$$

where the first, second, and third summands account for the inhibitory, cost, and bias connections, respectively.

For a densely connected symmetric neural net of size mn, the number of connections is

$$\frac{m^2n^2-mn}{2}+mn \ . \tag{13}$$

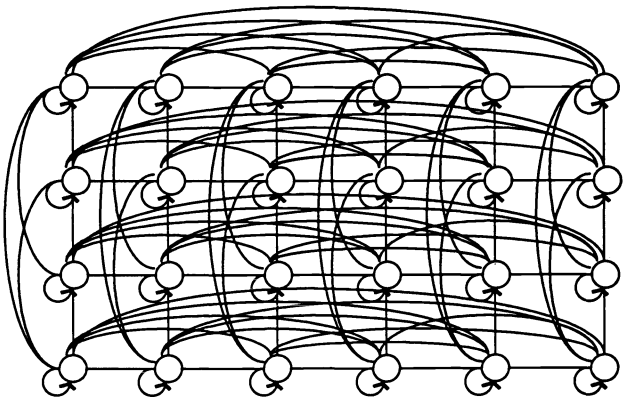


Fig. 4 Wiring diagram.

Taking a ratio of these expressions after factoring out common terms, yields

$$\frac{\frac{nm}{2}[(m-1)+(n-1)+2]}{\frac{nm}{2}[mn+1]} \quad . \tag{14}$$

If $\#c_s$ represents the number of connections in the neural clusterer and $\#c_d$ represents the number of connections in a densely symmetric neural network having mn neurons, then simplifying the preceding expression yields

$$\frac{\#c_s}{\#c_d}=\frac{m+n}{mn+1} \quad . \tag{15}$$

The advantage in the architecture of the neural clusterer over some hypothetical clusterer of the same size that is densely connected is most clearly evidenced in problem sizes where both m and n are large. Given that $m+n=C$, the efficiency of the sparsely connected neural clusterer is most evidenced as both m and n approach $C/2$.

3 Neural Properties

It will be shown that the network will converge to a point in neural space that is feasible and, with respect to the cost function described earlier, optimal. To this end, it is first demonstrated that optimal convergence with respect to the consensus function guarantees convergence to semifeasible or feasible configurations. We then show that optimization of the consensus function is equivalent to optimization of the cost function. Using these results, an argument is then presented illustrating that optimization of the cost function is the driving force that constrains the network to produce feasible solutions. This development is based on techniques presented in Ref. 17 using variations of that approach to account for $f(X)$ and the associated neural architecture.

The connection strengths of the neural clusterer are defined as follows: for all $\{e_{ab},e_{ab}\}$ in C_b,

$$s_{\{e_{ab},e_{ab}\}} > \sum_{y=0}^{y=n-1} c_{by} \quad ; \tag{16}$$

for all $\{e_{ab},e_{cb}\}$ in C_{in},

$$s_{\{e_{ab},e_{cb}\}} < -s_{\{e_{ab},e_{ab}\}} = -s_{\{e_{cb},e_{cb}\}} \quad ; \tag{17}$$

and for all $\{e_{ab},e_{ad}\}$ in C_c,

$$s_{\{e_{ab},e_{ad}\}} = -c_{bd} \quad . \tag{18}$$

Let S_γ be the set of configurations for which at least one sighting is in at least two GARCs and let S_η be the set of configurations for which there is at least one sighting that is in no GARC. For all configurations,

$$k \in S_\gamma \cup S_\eta \quad , \tag{19}$$

it will be shown that there exists at least one neuron in k such that when that neuron's state is changed, the energy of the neural network increases. This will prove that any locally optimal configuration cannot have empty columns or columns with more than one processing element on. Thus any locally optimal configuration is either feasible or semifeasible.

If k is in S_γ, then there exists a column that has at least two neurons on. Choose any one of these neurons, e_{ab}, and turn it off. Because the sum of the negative-valued inhibitory and cost connections outweigh the positive-valued bias connection of e_{ab}, turning it off increases the energy of the clusterer.

The following mathematically illustrates this point. Let $\Sigma_{c/\text{on}}$ represent the sum of the costs between the sighting represented by e_{ab} and all other sightings that are on in row a and let $\Sigma_{c/\text{tot}}$ represent the sum of the costs between the sighting represented by e_{ab} and all other sightings. We denote the bias of the neuron by

$$s_{\{e_{ab},e_{ab}\}} = \partial_b + \sum_{c/\text{tot}} \quad , \tag{20}$$

where

$$0 < \partial_b \quad . \tag{21}$$

Further, represent each inhibitory connection between e_{ab} and members in column b as

$$s_{\{e_{ab},e_{cb}\}} = -\partial_{\text{in}} - s_{\{e_{ab},e_{ab}\}} = -\partial_{\text{in}} - \partial_b - \sum_{c/\text{tot}} \quad , \tag{22}$$

where

$$0 < \partial_{\text{in}} \quad . \tag{23}$$

Thus, the amount of energy contributed by e_{ab} to the total energy of the neural clusterer is

$$\mu\left(-\partial_{\text{in}} - \partial_b - \sum_{c/\text{tot}}\right) - \sum_{c/\text{on}} + \partial_b + \sum_{c/\text{tot}} \quad , \tag{24}$$

or alternatively,

$$\partial_b(1-\mu) - \mu\partial_{\text{in}} + (1-\mu)\sum_{c/\text{tot}} - \sum_{c/\text{on}} \quad , \tag{25}$$

where μ represents the number of processing elements on in column b other than e_{ab}. Note that even if μ is 1, which minimizes the total number of negative connections activated in column b given that $k \in S_\gamma$, and e_{ab} is the only processing element on in row a, which minimizes the total negativity

contributed by the cost connections, then the total contribution to the energy resulting from e_{ab} is still negative due to the inhibitory factor ∂_{in}. The terms ∂_b and ∂_{in} may vary in each column for experimental reasons. For simplicity, they do not in this exposition nor in any of the programs run thus far.

If k is in S_η, then there exists a column b that has no neurons on. Choose any processing element in that column, say e_{ab}, and turn it on. Because the bias of e_{ab} is greater than the sum of all possible costs incurred in row a and because there are no inhibitory connections activated in column b, the total energy of the clusterer goes up. That is to say, the increment added to the total energy resulting from the activation of a processing element e_{ab} is given by setting μ equal to 0 in expression (25), yielding

$$\partial_b + \sum_{c/\text{tot}} - \sum_{c/\text{on}} , \tag{26}$$

which is strictly greater than 0. This completes the argument that any optimal configuration is either feasible or semifeasible.

In any semifeasible configuration, if one of the elements that is off is turned on, then the energy of the system goes down by an amount

$$\partial_{\text{in}} + \sum_{c/\text{on}} , \tag{27}$$

which is the magnitude of expression (25) with μ equal to 1.

If one of the elements that is on is turned off, then expression (26) gives the magnitude of the resulting energy decrement of the system. (This argument shows that semifeasible configurations are locally maximal. It applies as well to feasible configurations.) Thus the simple types of energy increasing arguments discussed earlier cannot be used to show convergence to feasible configurations. This point is returned to later.

We conform to the consensus function notation presented in Ref. 18 and express the value of processing element e_{xy} in configuration k (1 for on and 0 for off) as $k(e_{xy})$. For all configurations satisfying the constraints included in the definition of the cost function, the consensus function takes the form

$$C(k) = \sum_{\{e_{ip}, e_{ip}\} \in C_b} s_{\{e_{ip}, e_{ip}\}} k^2(e_{ip}) + \sum_{\{e_{ip}, e_{iq}\} \in C_c} s_{\{e_{ip}, e_{iq}\}} k(e_{ip}) k(e_{iq}) , \tag{28}$$

where $0 \leq i \leq m-1$ and $0 \leq p,q \leq n-1$.

Because the bias of a processing element depends only on the column and because in a feasible or semifeasible configuration, exactly one processing element must be on for every sighting, Eq. (28) becomes

$$\begin{aligned} C(k) &= n\sum_{c/\text{tot}} + n\partial_b + \sum_{\{e_{ip}, e_{iq}\} \in Cc} s_{\{e_{ip}, e_{iq}\}} k(e_{ip}) k(e_{iq}) \\ &= K + \sum_{\{e_{ip}, e_{iq}\} \in C_c} s_{\{e_{ip}, e_{iq}\}} k(e_{ip}) k(e_{iq}) \\ &= K - \sum_{\{e_{ip}, e_{iq}\} \in C_c} c_{\{p,q\}} k(e_{ip}) k(e_{iq}) , \end{aligned} \tag{29}$$

where n is the total number of sightings. This means that the cost function is minimal at configuration k if and only if the consensus function is maximal at k. Because the acceptance criteria for a state change is given by

$$A_k(u,c) = \frac{1}{1 + 1/\{\exp[\Delta C_k(u)/c]\}} , \tag{30}$$

where $\Delta C_k(u) = C(k_u) - C(k)$ and k_u is defined as the configuration that is obtained from configuration k by changing the state of processing element u, theory dictates that the neural network converges to a maximal or near maximal consensus value and hence to a minimal or near minimal cost value.

It can now be argued that the locally optimal points that the network converges to are feasible. We have just demonstrated that the network converges to configurations k, where

$$k \notin S_\gamma \cup S_\eta , \tag{31}$$

and seeks to globally minimize the expression

$$\sum_{\{e_{ip}, e_{iq}\} \in C_c} c_{\{p,q\}} k(e_{ip}) k(e_{iq}) \tag{32}$$

over all configurations. This means that the network strives to restrict the number of on elements in each column in the neural array to 1, while simultaneously trying to minimize expression (32). By examination, the minimization of expression (32) involves two factors:

1. the efficient arrangement of 1-valued processing elements among the rows of the neural array (i.e., keeping sightings with high cost values out of the same row)
2. the decoupling of pairs of processing elements that are on in the same row. This results from the fact that nonzero contributions to the sum can be made only by pairs of processing elements that are on in the same row.

It is factor 2 that inherently drives the network to utilize any zero row, causing it to have processing elements turned on in that row. Thus expressions (31) and (32) together imply that the network converges to optimal feasible solutions.

In all of the computer simulations run thus far, no locally optimal configuration reached has ever had a row with all 0-valued processing elements.* Even when m approached the value of n, all rows had 1-valued processing elements.

To maintain the structural simplicity of the neural network, a slight glitch in the cost function representation was allowed. The cost function does not take into account an averaging procedure among members of the same GARC. For example, as represented in Fig. 5, sightings s_1, s_2, and s_3 are in GARC G_1 and sightings s_4 and s_5 are in GARC G_2. Assume s_x is to be grouped in either GARC G_1 or G_2. Then, in theory, the

*It is interesting to note that in the Boltzmann machine treatment of the traveling salesman problem,[10] the preclusion of an empty row *and* column (indicating the exclusion of a city in the computed cost value at that point in time of the neural net's processing) was enforced by choosing the connection strength values in a certain manner. This was necessary because the corresponding neural network was seeking to minimize the path length value and exclusion of cities under consideration was consistent with this goal.

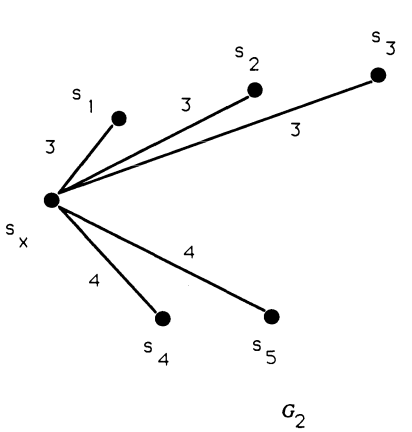


Fig. 5 Cost glitch.

neural clusterer will place s_x in G_2 because the inclusion of s_x in G_1 would raise the cost by 9 and the inclusion in G_2 would only raise the cost by 8. In principle, s_x should be grouped in G_1 because the average cost to its members is 3 and the average cost to the members of G_2 is 4. A situation representing this grouping phenomenon occurred in only one case and was obvious enough for a user to see. When the tracking team improved the cost functions, this misgrouping did not occur. Because of the rarity of this occurrence and the fact that an averaging mechanism could add substantially to the length of the convergence process, a resolution to the problem was not pursued.

4 Annealing Schedule

We use the parallel cooling schedule as described in Ref. 18. If we denote all connections to processing element x (aside from the connection from x to x) by Syn_x, the cooling schedule start value $c_{0,e}$ for each processing element e is given by

$$c_{0,e}=\sum_{\{x,y\}\in \mathrm{Syn}_x}|s_{\{x,y\}}| \ . \tag{33}$$

If we represent the change in the consensus function resulting from the change of processing element x in configuration k, $\Delta C_k(x)$, by

$$[1-2k(x)]\left[\sum_{\{x,y\}\in \mathrm{Syn}_x} s_{\{x,y\}}k(y)+s_{\{x,x\}}\right] , \tag{34}$$

then

$$\Delta C_k(x)/c_{0,e}=\frac{[1-2k(x)]\left[\sum_{\{x,y\}\in \mathrm{Syn}_x} s_{\{x,y\}}k(y)+s_{\{x,x\}}\right]}{\sum_{\{x,y\}\in \mathrm{Syn}_x}|s_{\{x,y\}}|}$$

$$=[1-2k(x)]\left[\Delta_1+\frac{s_{\{x,x\}}}{\sum_{\{x,y\}\in \mathrm{Syn}_x}|s_{\{x,y\}}|}\right] , \tag{35}$$

where

$$\Delta_1\leq 0 \text{ and } |\Delta_1|\leq 1 \ . \tag{36}$$

Expanding the numerator and denominator of the fraction yields

$$s_{\{x,x\}}=\partial_x+\sum_{c/\mathrm{tot}} , \tag{37}$$

and

$$\sum_{\{x,y\}\in \mathrm{Syn}_x}|s_{\{x,y\}}|=\left|\rho\left(-\partial_{\mathrm{in}}-\partial_b-\sum_{c/\mathrm{tot}}\right)-\sum_{c/\mathrm{on}}\right| , \tag{38}$$

where $\rho=m-1>0$ because m is assumed to be greater than 1 and $\Sigma_{c/\mathrm{tot}}$ represents the sum of the costs between sighting x and all other sightings. This yields

$$\Delta C_k(x)/c_{0,e}=[1-2k(x)]\left[\Delta_1+\frac{\partial_x+\sum_{c/\mathrm{tot}}}{\rho\left(\partial_{\mathrm{in}}+\partial_x+\sum_{c/\mathrm{tot}}\right)+\sum_{c/\mathrm{tot}}}\right]$$

$$=[1-2k(x)](\Delta_1+\Delta_2) , \tag{39}$$

where

$$0<\Delta_2<1 \ . \tag{40}$$

This implies that

$$0\leq|\Delta C_k(x)/c_{0,e}|=|1-2k(x)|\,|\Delta_1+\Delta_2|<1 \ . \tag{41}$$

Given this, we are guaranteed that the generating probability is such that

$$0.2689414<A_k(u,c_{0,e})<0.7310585 , \tag{42}$$

yielding a sufficient amount of initial randomness for simulated annealing type searching. The fact that initial states are randomly generated indicates that the initial generating probabilities are somewhat distanced from the indicated upper and lower bounds.

5 Graphics Interface

Underlying the relationship between the neural configuration and its corresponding visual analogue at a prescribed time is a mapping that uses information relating to the distribution of the n sightings among the m GARCs to form multicolored lines between sighting locations on a screen.†

More precisely, given that n sightings are to be appropriately subdivided among m GARCs, and given a particular configuration of the $m\times n$ neural array, as shown in Fig. 2, then a line of color x is drawn between two sightings with detection times t_1 and t_2 if the neural configuration places

†It was desired to set up a correspondence between the configuration and the graphical representation so that as the neural net approached a feasible solution, the graphics would approach a great circle arc grouping of sightings. This would not only provide for the correct visualization of final groupings of sightings into GARCs but would also allow the user to see various grouping patterns long before the network was done with its visual convergence.

Order of Detections	1	2	3	4	5	6	7	8	9
Sighting Numbers	2	6	1	8	5	3	4	9	7

Time Ordering of Sightings

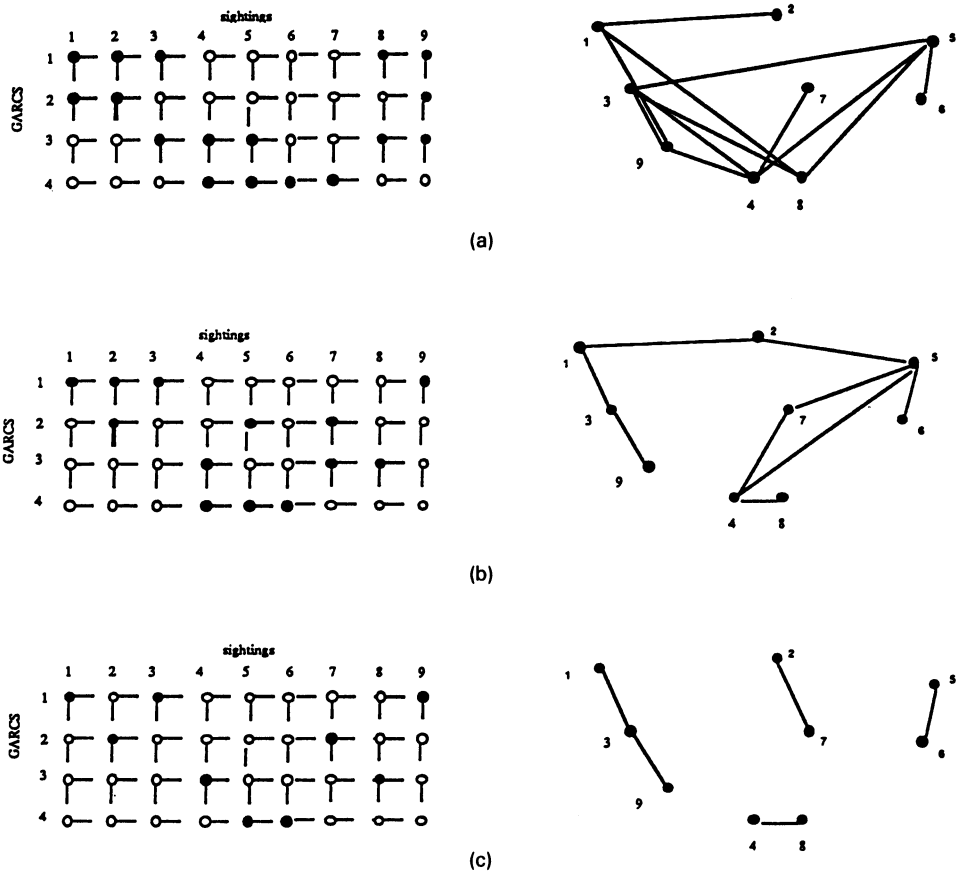


Fig. 6 Configuration/graphics sequence.

them in the same GARC, the color associated with that GARC is x, and there is no other sighting in the GARC that was sighted within the time interval (t_1,t_2). At this point in the graphics development, if the same two (or more) sightings are in more than one GARC, the color of the connection(s) between the sightings corresponds to the GARC in which it was most recently decided the set should belong. In the future, these types of connections will be indicated either by a special coloring denoting association between sighting pairs in more than one GARC or by thickness, if color is not available. An example of a sequence of configuration-visualization pairs together with the time of detection orderings of the sightings is given in Fig. 6. The time orderings are important in the construction of the visual images. This can be seen, for example, in configuration/visualization pair (a) where in GARC 1 the sightings 1, 2, 3, 8, and 9 are connected in the order 2, 1, 8, 3, and 9, or in pair (b), where 4, 7, and 8 in GARC 3 are connected in the order 8, 4, and 7.

Figures 7 through 13 represent monochromatic versions of the tracker and neural graphics. Figure 7 represents the format of the screen related to neural net activity. The lower half of the screen represents the geographical perspective on which sightings will be overlaid when their locations are received from the tracker. The display screen can be zoomed in on any earth surface location using a variety of geographical projections. The projection shown here is Polar Azimuthal Equidistant (North American Rectangular is an alternate name used by the authors for purposes of categorization).

The upper left-hand screen represents the configuration of the neural network, as shown in Fig. 2 and described earlier in the paper. The upper right-hand side is the neural control panel. The first row of the panel allows variable sighting numbers to be entered. The cost of associating the two sightings is then given. The LABELS command of the next row causes the sighting number to appear next to its dot in the lower half of the screen. The STOP command halts the execution of the network. The rate of temperature decrease is given by

$$c_{0,e} * (\text{b\_par})^w \ , \tag{43}$$

where the value of b_par is designated in the neural net input parameters portion of the control panel. When the network is started, the value of w is 0. If the network is stopped, REINIT restarts the network at the last state reached by the network in its previous run. The value of w is automatically set to 0 so as to maximize initial randomness in search.

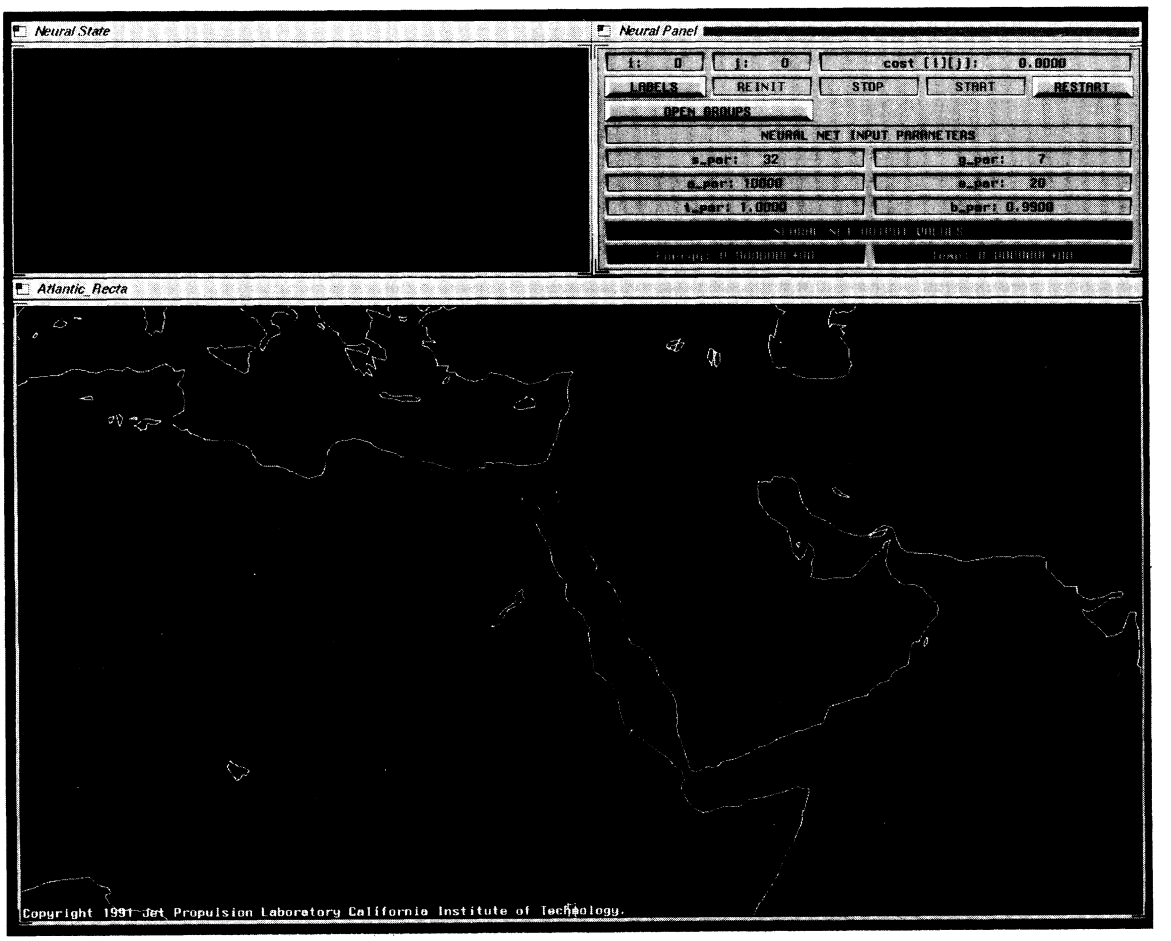


Fig. 7 Neural graphics display.

START initiates the network with any previous value of w reached and with the last neural state obtained. Whenever the network is stopped, the START command can be invoked. RESTART begins the network with $w=0$ and a random initial state. It can be invoked any time the network is stopped.

Parameters, as shown in the middle portion of the panel, can be altered and entered any time the network is stopped. These together with the various commands on the LABELS line of the control panel enable the user to interact with the running of the network and experiment with different annealing schedules. The input parameters s_par and g_par refer to the number of sightings and groupings, respectively. Variable a_par is related to how long the network is allowed to run, e_par denotes the length of time spent converging at the same temperature, where t_par is an adjustment concerning the randomness of search in the early stages of the neural processes and, as mentioned earlier, b_par determines the granularity of temperature decrease in the underlying simulated annealing search process.

The energy value at the bottom of the panel is a measure of how well the network has grouped the sightings. The higher the energy, the smaller the value of the cost function we are trying to minimize. The temperature parameter indicates a "measure of randomness of search" for the optimal solution. The higher the temperature, the more even the odds are that each neuron will accept a 0 or 1 state when it is given its turn at making a potential change. The lower the temperature, the more likely the neuron will choose the value that helps minimize the cost (or, equivalently, maximize the energy).

At any point in time during the running of the network, only the best neural configuration obtained is displayed. If several iterations of temperature decrease occur without a graphics display change (indicating a better state was not reached), or if the temperature is less than 10^{-4}, then the displayed state is defined to be the final state of the network. Experiments conducted thus far indicate that the state displayed under these conditions is the final state reached by the network.

Figure 8 represents a sample of tracker output. The crosses represent the sightings and the lines represent the tracks formed by the tracker to connect the sightings. The user selects one or more of the available tracks formed by the system. The selected tracks are indicated by the solid lines intersecting circles. The heavier lines at the various corners of the chosen tracks represent the assumed maneuver taken by the aircraft forming the track. The total number of sightings in the selected tracks together with the numbers of GARCs comprising these selected tracks is entered into the tracking panel, as shown in the middle left of the figure.

These parameters together with the geographical location of the sightings associated with the selected tracks are then sent to the neural clusterer. The remaining parameters of the

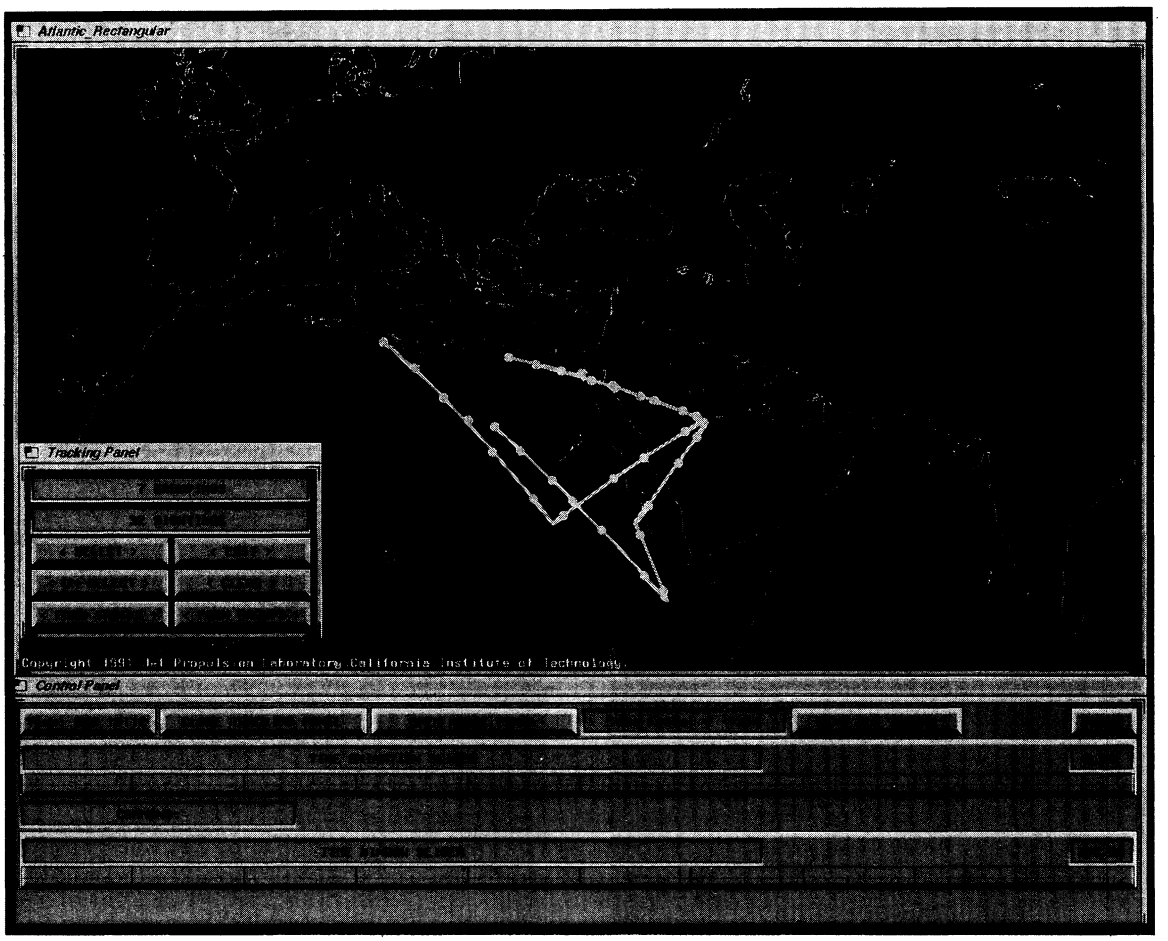


Fig. 8 Tracker display—Riyadh scenario 1.

neural control panel have default values that can be altered or left alone as the user wishes.

The neural control panel is used for research and development purposes. However, simple heuristics learned from experimentation could be provided so that a relatively inexperienced user could improve the results of the system in both speed and quality.

Figures 7 through 13 serve as a dramatic illustration of the potential of neural networks for rapidly and correctly finding GARCs. In all tracker output diagrams represented in this paper, there was a 98% match between the tracker output and the real world. Because the tracker and neural clusterer agreed in these cases, the clusterer obtained accuracies at such a level as well. Although it was not rigorously demonstrated that the solutions reached by the network in these cases was optimal in the global sense, globally optimal solutions were obtained by the clusterer architecture in all cases for which non-neural determination of the global optimal was relatively straightforward.

Figure 9 concerns a scenario near Riyadh, Saudi Arabia. Figure 9(a) represents the unconnected sightings as sent to the neural clusterer by the tracker. Figure 9(b) represents a randomly generated initial state, whereas Figs. 9(c) and 9(d) are mid and final states of the clusterer. These represent only a few of the many different states displayed by the neural graphics during the course of its convergence.

Figure 10 represents the same scenario as Fig. 8 except that a different subset of sightings is chosen to be sent to the clusterer. Figure 11 illustrates the convergence sequence for the sightings corresponding to these tracks.

Figure 12 illustrates the tracks formed by aircraft over Norway. Figure 13 gives a convergence sequence for the chosen track. Here, the sighting identifications were chosen so that consecutive numbering appeared in the same GARC. This arrangement more readily conveys the configuration mechanics relating to the gradual but eventual success of the underlying neural network in obtaining proper clusters.

One question of interest in examining the utility of Boltzmann machines for cost function optimization is how consistent their results are when they are started from different initial configurations. Figure 14 gives such a comparison for 10 different randomly chosen initial states, where b_par and e_par values are 0.9 and 10, respectively. Figure 15 gives convergence statistics for the same initial states (in the same order) with b_par and e_par equaling 0.95 and 15. Note that increasing the values of these parameters increases the likelihood of obtaining better solutions, but the price is paid for with longer convergence times. This motivates our investigation into a parallelization of the neural architecture. In most cases, the visual differences among final states with unequal energy values were minor and become even less significant as the annealing schedules become more finely grained. This

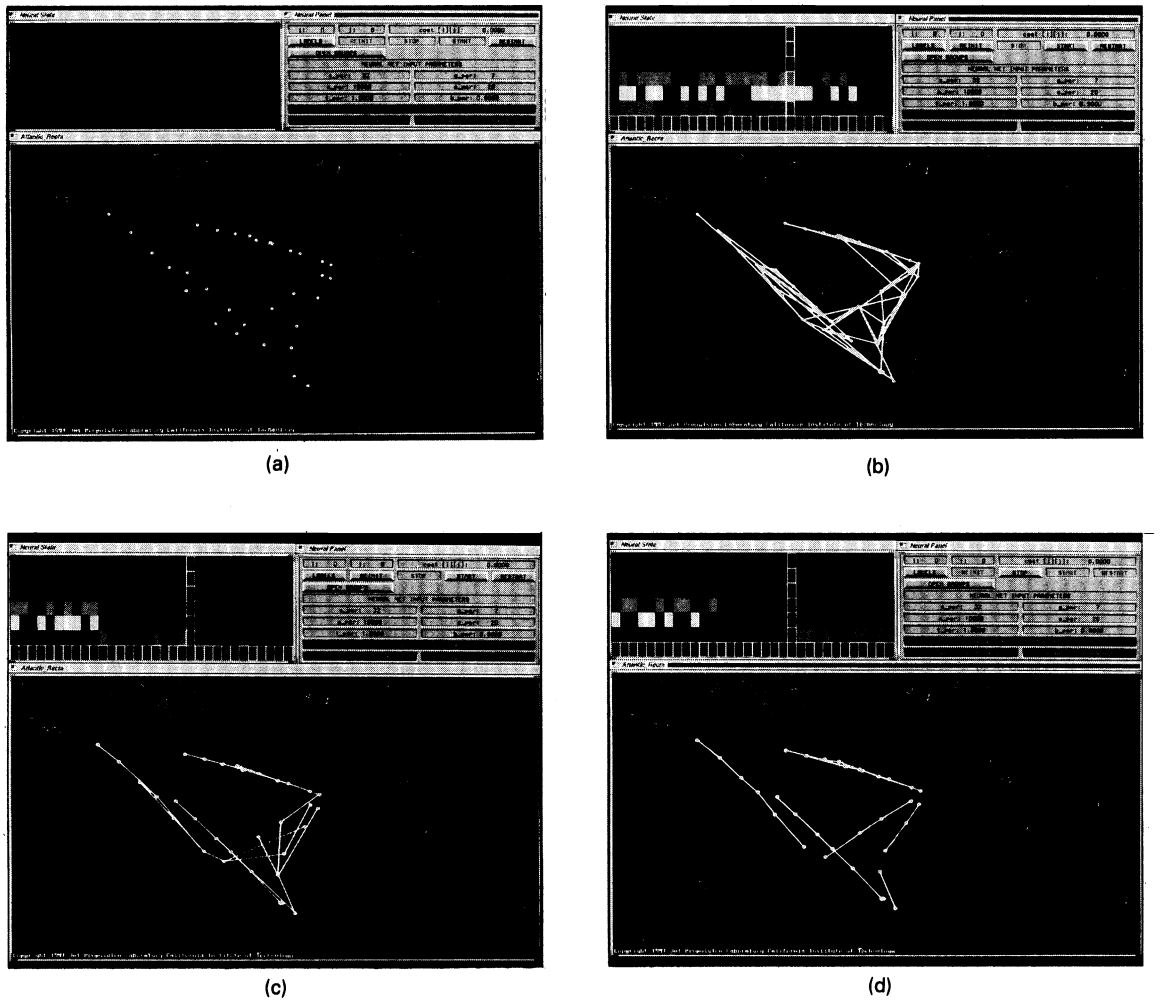


Fig. 9 Convergence sequence—Riyadh scenario 1: (a) unconnected sightings, (b) connections from randomly generated initial state, (c) midstate of clusterer, and (d) final state of clusterer.

is evidenced by comparing the values of σ_ξ in Figs. 14 and 15.

In all runs having neural sizes comparable to the ones displayed, results were of the same level in terms of speed and quality.

It is interesting to compare the number of steps needed by the neural architecture to arrive at the correct grouping of the sightings according to the prescribed cost function as compared to a combinatorially exhaustive approach. If there are n sightings to be divided into m GARCs, we define a candidate grouping to be

$$[n_1,\ n_2,\ n_3,\ \ldots,\ n_m]\ , \tag{44}$$

where

$$n_1 \geq n_2 \geq n_3 \geq \cdots \geq n_m > 0 \quad \text{and} \quad \sum_{\nu=1}^{m} n_\nu = n\ . \tag{45}$$

If Ψ represents the set of all candidate groupings and Ξ is the set of the total number of possible groupings of n sightings into m clusters, we have

$$|\Xi| = \sum_{[n_1,n_2,n_3,\ldots,n_m]\in\Psi} \binom{n}{n_1}\binom{n-n_1}{n_2}\binom{n-n_1-n_2}{n_3}\cdots\binom{n-\zeta(n_\nu,m)}{n_m}\ , \tag{46}$$

where

$$\zeta(n_\nu,m) = \sum_{\nu=1}^{m-1} n_\nu\ . \tag{47}$$

Figure 16 represents a comparison of the number of steps taken by the neural network to obtain the final state as opposed to a minimal bound on the number of steps needed to obtain the best grouping using an exhaustive approach. Under the designation of each scenario is a randomly chosen candidate grouping corresponding to that scenario's n and m values. The value w relates to the neural scenario and represents the approximate number of temperature decreases until the final state is displayed. Thus total number of neural iterations n_i

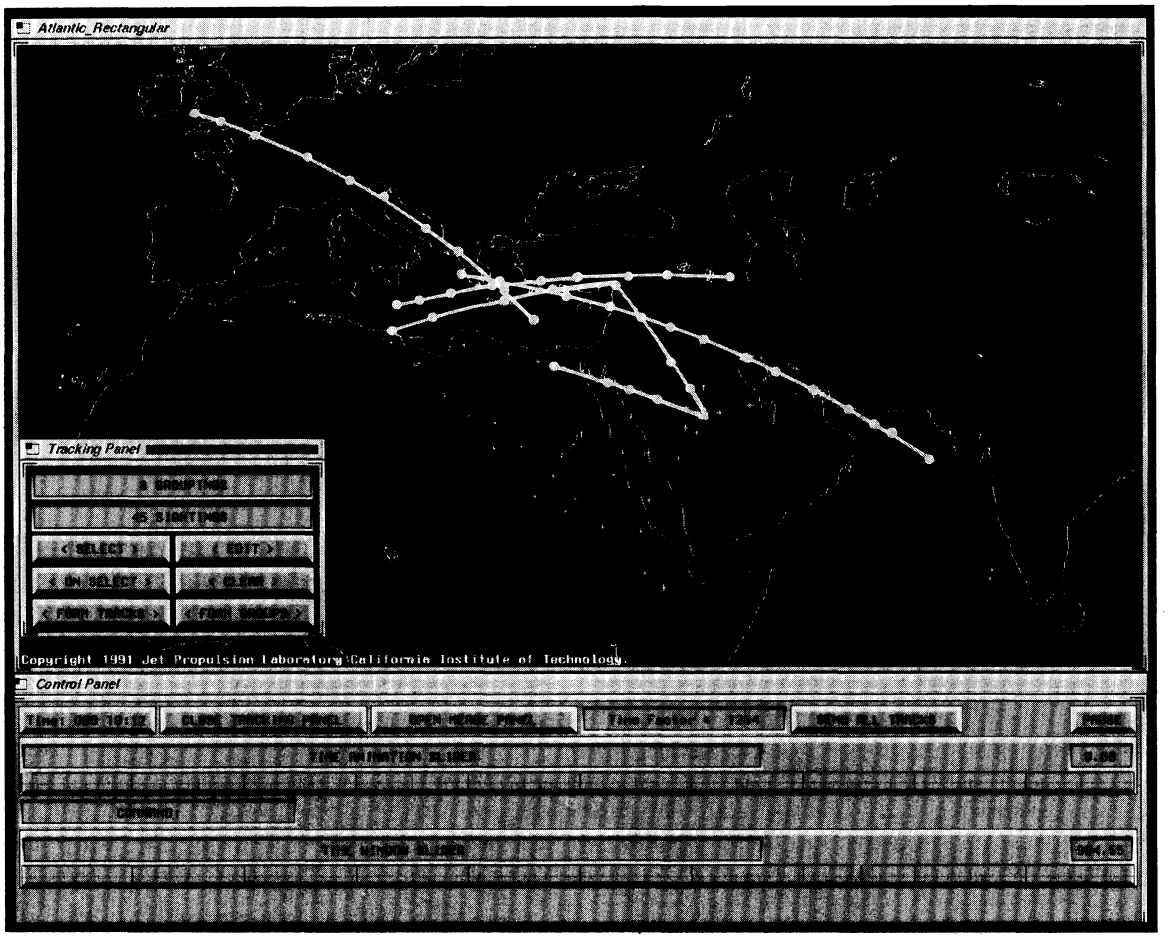


Fig. 10 Tracker display—Riyadh scenario 2.

needed by the Boltzmann machine to obtain the final graphically displayed state is given by

$$n_i = n * m * w * \text{e\_par} \ . \tag{48}$$

To see how this number compares approximately to the total number of clustering possibilities, we compute only the first binomial factor of that scenario's candidate grouping. The resulting number, designated by Λ, is only a fraction of the total set $|\Xi|$. The variable n_i^p represents the number of sequential steps taken by the neural network if all neurons are allowed to update simultaneously. As can be seen in the figure,

$$n_i^p \ll n_i \ll \Lambda \ll |\Xi| \ . \tag{49}$$

Each neural iteration involves (1) the determination of $\Delta C_k(x)$, which requires an addition and multiplication operation with each of $n-1+m-1$ other elements of the neural network together with an addition of a bias term; (2) determination of acceptance probability $A_k(u,c)$; (3) generation of a random number; (4) determination as to whether a state change is to be made based on a comparison of $A_k(u,c)$ with the random number; and (5) potential implementation of the state change. The total number of computations in steps 2, 3, 4, and 5 is approximately 40 (assuming 20 scenario-independent computations for the simple yet effective random number generator used and 14 computations for the exponential function computation to the fourth decimal place). The approximate total number of computations per neural iteration is therefore

$$2(n-1)+2(m-1)+40+5 \ , \tag{50}$$

where the last summand represents the addition of the first two terms and the bias and the determination of the sign of $\Delta C_k(x)$. (This estimate is somewhat inflated because as the network converges, many neural values represented in this computation are 0.) Correspondingly, each member of Ξ must have its sum computed, which involves of the order of

$$\frac{n_1(n_1-1)}{2}+\frac{n_2(n_2-1)}{2}+\ldots+\frac{n_m(n_m-1)}{2}+m-1 \tag{51}$$

operations, where the last summand represents the addition of the first m terms. For the groupings chosen in Fig. 16, the number of operations per neural iteration either approximates that required for the corresponding combinatorial sum evaluation or is substantially less than that number. As the number of sightings is more evenly distributed among the groupings, it becomes more likely that the number of operations per neural iteration will supersede the number of operations in

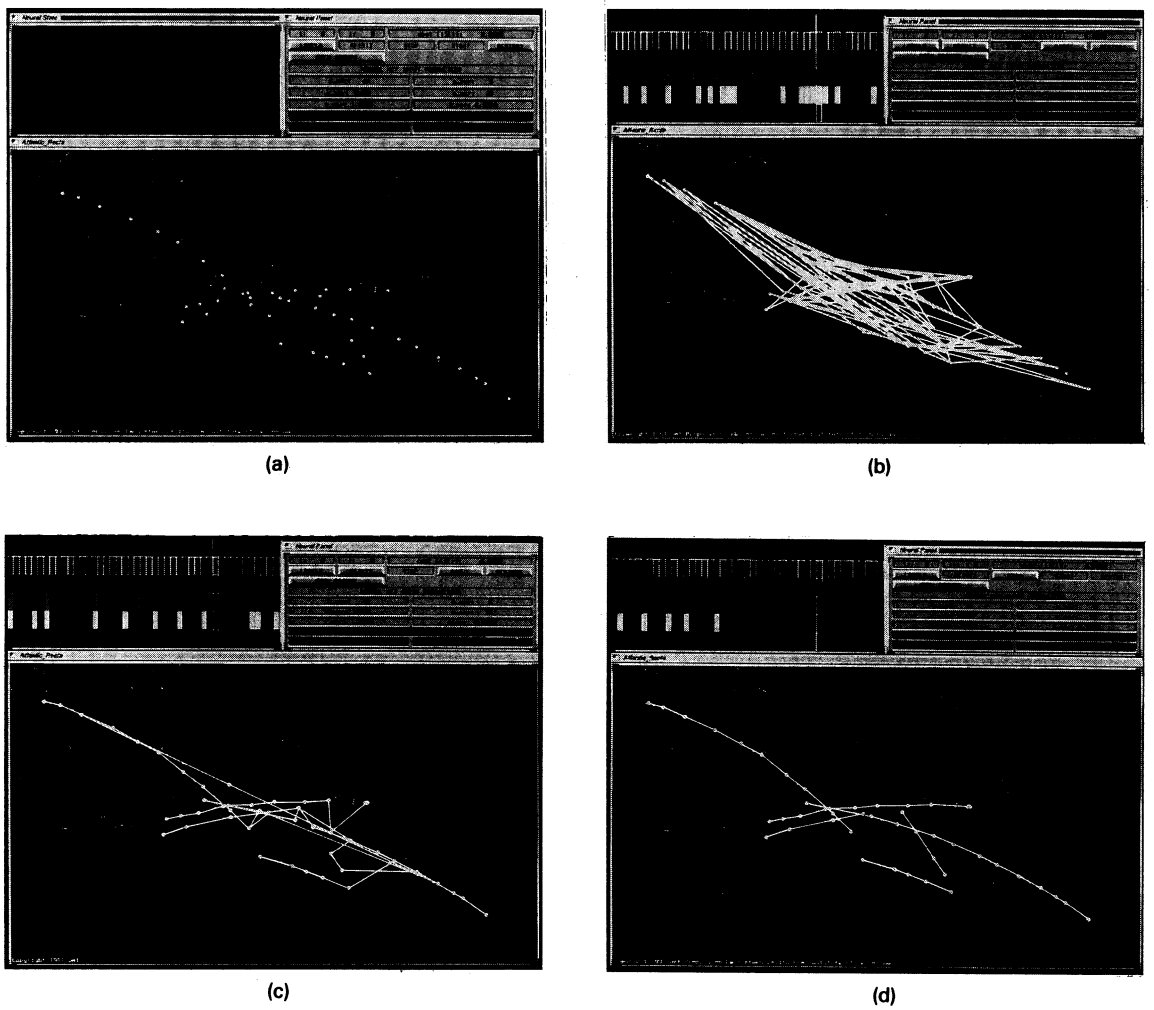


Fig. 11 Convergence sequence—Riyadh scenario 2: (a) unconnected sightings, (b) connections from randomly generated initial state, (c) midstate of clusterer, and (d) final state of clusterer.

the computation of the combinatoric evaluation of the sum. It is expected, however, that the cardinality of Ξ will counterbalance this and ensure the overall efficiency of the neural approach over the exhaustive combinatoric search approach.

Simple steps could be taken to reduce the number of additions per each member of Ξ. Further, as m approaches n, the number of computational operations per neural iteration supersedes the number of operations per each member of Ξ. It is questionable whether the consideration of these factors together or separately would be enough to counterbalance the comparatively large cardinality of Ξ. It is further questionable whether a parallel design of a combinatorial approach for arbitrary values of n and m would be burdened by communications as compared to a parallel software, hardware, or optical neural implementation, which inherently favors decomposition. Further experiments are needed relating the efficiencies in these cases.

6 Future Work

It is desired to find a fixed set of values for b_par and e_par that would guarantee quick as well as near-real-world convergence for all data sets. Typically, excellent GARC formation for data where $nm > 1000$ requires values of b_par $\leqslant 0.99$ e_par $\geqslant 20$. Given this, the potential for neural net treatment of the clustering problem has been demonstrated.

The convergence times for many cases using these parameters are of the order of minutes. We will experiment with the lean architectural design to produce a parallel decomposition of the net on varying computers to produce faster convergence times. Further, increasing the potential for user interaction will decrease the need for the program to achieve optimal or near optimal solutions (the absence of these requirements in general provides for much shorter convergence times). The user can then interact with the neural program to determine a more visually and computationally palatable neural convergence, while at the same time checking to see if the GARC decomposition solution by neural standards is consistent with that obtained by the tracker. Enhanced speed of convergence will also be obtained when typing or aircraft identification information is used in conjunction with kinematic criteria in determining cost values.

At present, the m value is a by-product of the tracker resulting from that system's own indirect GARC determination. An effort to determine the m value using the neural clusterer involves the determination of a range of plausible m values given the number of sightings and the associated

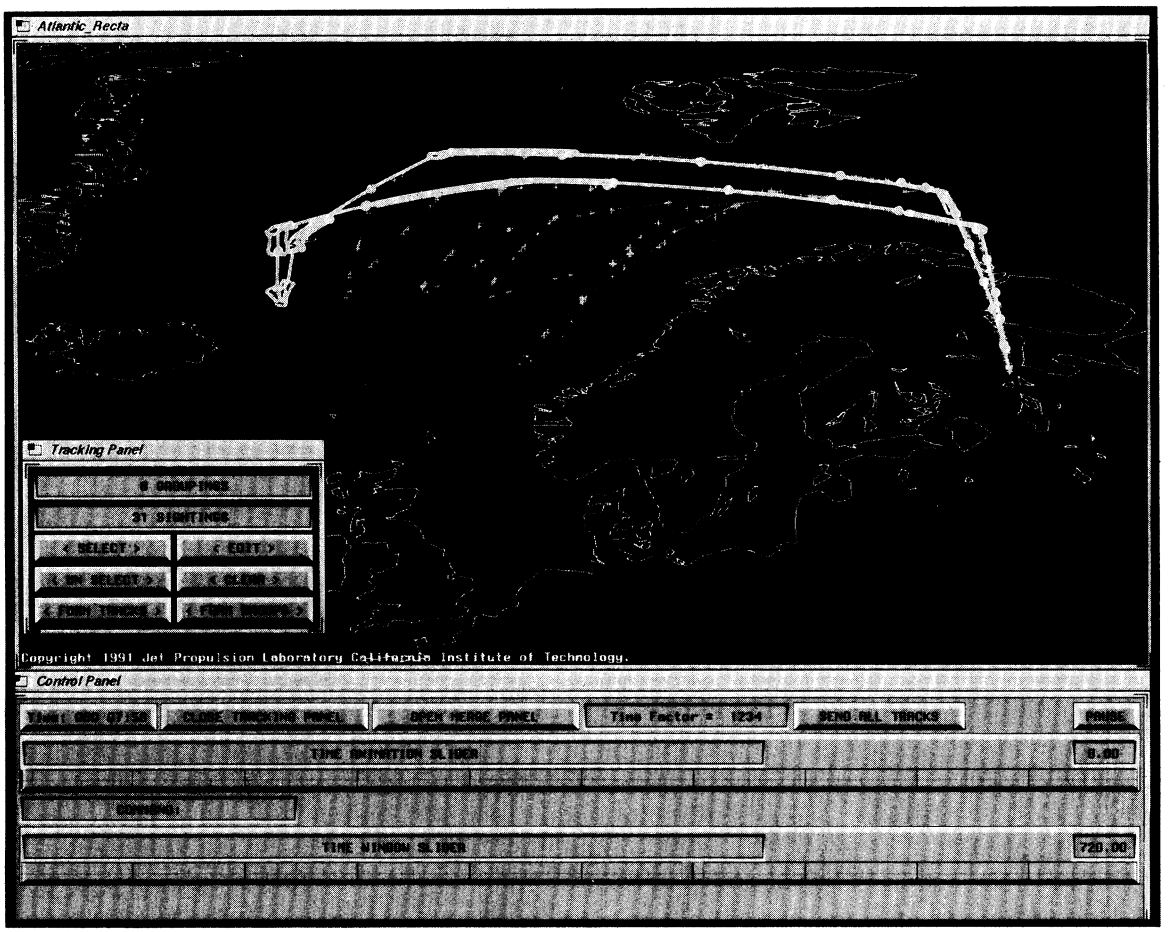


Fig. 12 Tracker display—Norway.

costs between sightings. Given this m range, several copies of the neural clusterer would operate in parallel. Each copy would work on a different m value and would itself be decomposed across several processors. Convergence properties of each of the neural nets would be used to determine which m value in the m range is correct. An algorithm providing for this determination characterizes the research effort in this area.

Plans for experimentation with more efficient digital and neural hardware are also in effect. As the feasibility of optical implementation of neural networks becomes more and more evident, the relevance of incorporating this medium with the clusterer architecture is becoming more and more apparent.

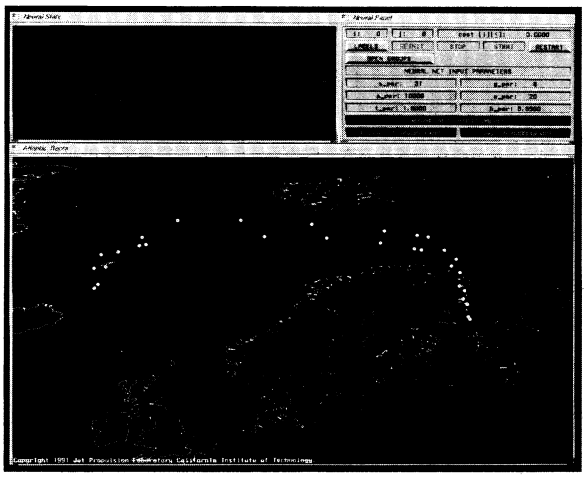
Other more conventional digital implementations of clustering algorithms exist based on the ideas in Ref. 19. We have found that there are cases in which these algorithms yield correct solutions and run reasonably faster than the present implementation of the neural clusterer. It is expected, however, that when the neural clustering effort is carried to its logical end (that is to say, when the neural design is implemented on some form of electrical or optical hardware), the Boltzmann design will immeasurably outperform these more traditional, inherently sequential clustering algorithms. Further, an analysis of these algorithms reveals that, in many cases, their inherently sequential structure yields incorrect clusterings in cases where the neural cluster gives correct solutions. Continued study of the nature of these data sets would give direction as to the determination of classes of scenarios on which the neural cluster would give superior performance even in its present nonparallel digitally implemented form.

7 Conclusions

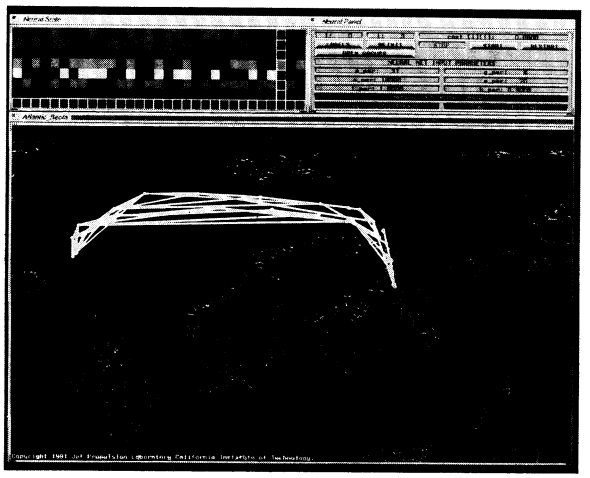
The GARC formations of the Boltzmann machine neural clusterer closely mirror those of the real world in virtually all executions. In many scenarios, the answers are obtained within a matter of minutes. To achieve rapid convergence where the numbers of sightings are of size 100 or greater, investigations into parallelization options, user manipulation of GARCs, and the development of improved cost values will be emphasized. The parallelization of the neural architecture will be facilitated by its sparsity.

Acknowledgments

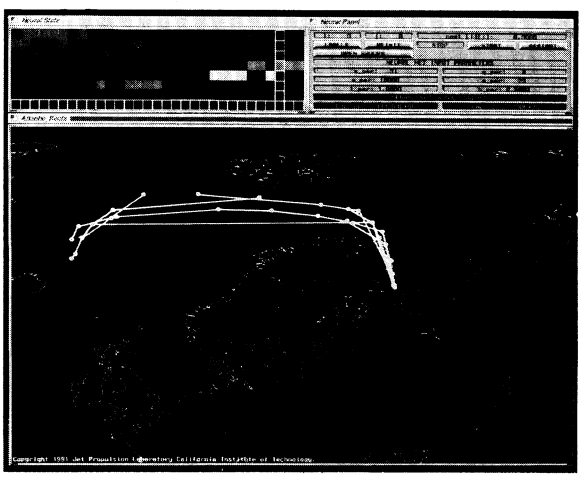
The authors would like to thank T. Gottschalk and M. Pomerantz for their technical support. Further appreciation goes to D. Curkendall, A. Loomis and H. Henry for their guidance and encouragement. The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.



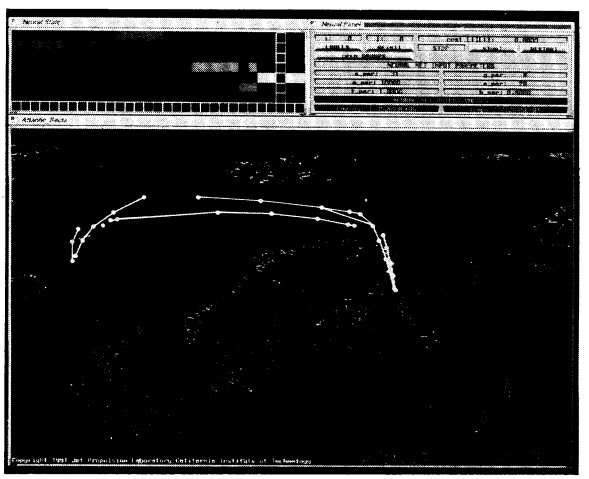
(a)



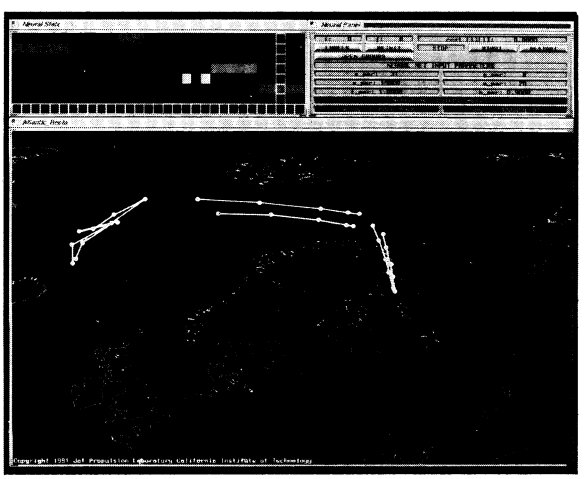
(b)



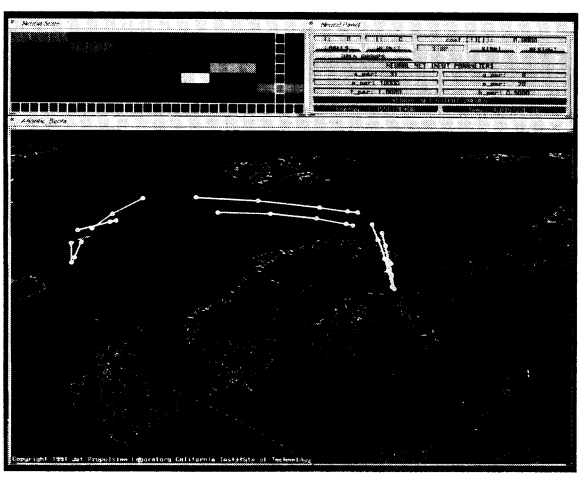
(c)



(d)



(e)



(f)

Fig. 13 Convergence sequence—Norway: (a) unconnected sightings, (b) connections from randomly generated initial state, (c) early convergence stage, (d) midconvergence stage, (e) latter convergence stage, and (f) final state of clusterer.

Riyadh / scenario 1		Riyadh / scenario 2		Norway	
Energy	*cpu time/secs*	*Energy*	*cpu time/secs*	*Energy*	*cpu time/secs*
1.279418E +06	30.71	8.723264E +06	52.08	1.455613E +06	41.96
1.278249E +06	31.55	8.723264E +06	58.27	1.455556E +06	44.25
1.279418E +06	27.50	8.723264E +06	62.80	1.455613E +06	42.72
1.279418E +06	32.28	8.723264E +06	57.32	1.455545E +06	49.12
1.279418E +06	32.58	8.723264E +06	55.31	1.455558E +06	42.5
1.279418E +06	33.37	8.716774E +06	59.43	1.455545E +06	53.92
1.279418E +06	30.48	8.723264E +06	56.38	1.455613E +06	45.54
1.278183E +06	32.20	8.723264E +06	58.88	1.455545E +06	56.48
1.279418E +06	33.48	8.723264E +06	55.35	1.455553E +06	51.04
1.279418E +06	30.72	8.723264E +06	58.35	1.455613E +06	43.57
$\bar{\xi}$	$\bar{\tau}$	$\bar{\xi}$	$\bar{\tau}$	$\bar{\xi}$	$\bar{\tau}$
1.279178E + 06	31.487	8.722615E + 06	57.417	1.455575E + 06	47.119
σ_ξ	σ_τ	σ_ξ	σ_τ	σ_ξ	σ_τ
481.0266	1.6720	1947.00	2.7395	31.00	4.9301

Fig. 14 Convergence statistics for b_par = 0.9 and e_par = 10.

Riyadh / scenario 1		Riyadh / scenario 2		Norway	
Energy	*cpu time/secs*	*Energy*	*cpu time/secs*	*Energy*	*cpu time/secs*
1.279418E +06	90.05	8.723264E +06	156.13	1.455613E +06	144.77
1.279418E +06	88.60	8.723264E +06	171.91	1.455556E +06	131.82
1.279418E +06	90.00	8.722095E +06	151.50	1.455613E +06	145.93
1.279418E +06	83.16	8.723264E +06	156.13	1.455613E +06	144.15
1.279418E +06	87.58	8.723264E +06	156.31	1.455613E +06	150.38
1.279418E +06	85.77	8.723264E +06	157.57	1.455613E +06	150.60
1.279418E +06	84.57	8.723264E +06	174.65	1.455613E +06	144.01
1.279418E +06	87.01	8.723264E +06	165.36	1.455613E +06	155.17
1.279418E +06	84.75	8.723264E +06	161.40	1.455613E +06	135.86
1.279418E +06	90.81	8.723264E +06	164.25	1.455553E +06	122.87
$\bar{\xi}$	$\bar{\tau}$	$\bar{\xi}$	$\bar{\tau}$	$\bar{\xi}$	$\bar{\tau}$
1.279418E +06	87.230	8.723147E +06	161.521	1.455601E +06	142.556
σ_ξ	σ_τ	σ_ξ	σ_τ	σ_ξ	σ_τ
0.0	2.4982	350.7	7.0980	23.4115	8.7005

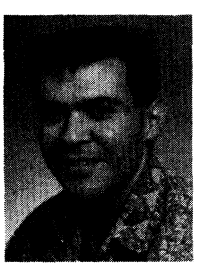
Fig. 15 Convergence statistics for b_par = 0.95 and e_par = 15.

	Riyadh / scenario 1 [18,5,3,2,2,1,1]		Riyadh / scenario 2 [15,10,6,5,5,4]		Norway [10,9,4,3,2,1,1,1]	
	b_par = .9 e_par = 10	*b_par = .95 e_par = 15*	*b_par = .9 e_par = 10*	*b_par = .95 e_par = 15*	*b_par = .9 e_par = 10*	*b_par = .95 e_par = 15*
n	32	32	45	45	31	31
m	7	7	6	6	8	8
w	75	157	86	183	91	218
n_1	168000	527520	232200	741150	225680	810960
Λ	471435600	471435600	> 3736E +08	> 3736E +08	44352165	44352165
n_1^p	750	2355	860	2745	910	3270

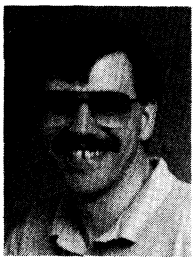
Fig. 16 Computational iteration comparison.

References

1. T. D. Gottschalk, "Tracking and correlation for space-based wide area surveillance," Aerospace Report No. TOR-0090 (5503-04)-9 (1990).
2. Private conversation with T. Gottschalk, Feb. 10, 1992.
3. T. D. Gottschalk, "Concurrent implementation of Munkres algorithm" in *Proc. 5th Distributed Memory Computing Conference, IEEE Computer Society Press,* **1,** 52–57 (April 1990).
4. R. M. Kuczewski, "Neural network approaches to multi-target tracking," *Proc. IEEE Int. Conf. Neural Networks,* pp. 347–361 (June 1987).
5. C. Priebe and D. Marchette, "An application of neural networks to a data fusion problem," in *Proc. 1987 Tri-Service Data Fusion Symp.* **1,** 226–235 (June 1987).
6. D. Sengupta and R. Iltis, "Neural solution to the multitarget tracking data association problem," *IEEE Trans. Aerospace Electron. Syst.* **AES25**(1), 96–108 (1989).
7. M. E. McCurry, "Neural network implementation of a scan-to-scan correlation algorithm," in *High Speed Computing, Proc. SPIE* **880,** 85–87 (1988).
8. G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: constraint satisfaction networks that learn," Technical Report No. CMU-CS-84-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa. (1984).
9. A. DeGloria, P. Faraboschi, and S. Ridella, "A dedicated massively parallel architecture for the Boltzmann machine," *Parallel Comput.* **18**(1), 57–73 (1992).
10. B. W. Lee and B. J. Sheu, "Paralleled hardware annealing for optimal solutions on electronic neural networks," *IEEE Trans. Neural Networks* **4**(4), 588–599 (1993).
11. B. Shirazi, "Neural nets, optical technology key current research areas, keynoter asserts," *Computer* **22**(9), 107–108 (1989).
12. H. Caulfield, J. Kinser, and S. Rogers, "Optical neural networks," *Proc. IEEE* **77**(10), 1573–1583 (1989).
13. D. Casasent and E. Barnard, "Adaptive clustering optical neural net," *Appl. Opt.* **29**(17), 2603–2615 (1990).
14. E. Barnard and D. Casasent, "Multitarget tracking with cubic energy optical neural nets," *Appl. Opt.* **28**(4), 791–798 (1989).
15. M. Yee and D. Casasent, "Multitarget data association using an optical neural network," *Appl. Opt.* **31**(5), 613–624 (1992).
16. J. Ohta, S. Tai, M. Oita, K. Kuroda, K. Kyuma, and K. Hamanaka, "Optical implementation of an associative neural network model with a stochastic process," *Appl. Opt.* **28**(12), 2426–2428 (1989).
17. J. H. M. Korst and E. H. L. Aarts, "Combinatorial optimization on a Boltzmann machine," *J. Parallel Distrib. Comput.* **6,** 331–357 (1989).
18. J. H. M. Korst and E. H. L. Aarts, *Simulated Annealing and Boltzmann Machines,* Wiley & Sons, New York (1989).
19. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data,* Prentice-Hall, Englewood Cliffs, N.J. (1988).



John Spagnuolo, Jr. has been a member of the technical staff at the Jet Propulsion Laboratory (JPL) at the California Institute of Technology since 1983. He graduated from Clarkson College of Technology and obtained an MA degree from the University of California at Los Angeles. In 1983 he received his PhD from the Rensselaer Polytechnic Institute in mathematical sciences. While at JPL, Dr. Spagnuolo has done work in artificial intelligence, parallel computation (time warp), coding theory, sensor modeling, simulated annealing, look-ahead planning, battle management, expert systems, neural networks, and automata theory. His present research interests include cognitive neurodynamics, neural automata, tensor analysis, and solar physics.



James B. Lathrop is a member of the technical staff at the California Institute of Technology's Jet Propulsion Laboratory where he has been involved in interactive computer graphics since 1985. He was educated in mathematics and physics at the University of Oregon where he received his BS degree in 1984. His recent interest is in merging interactive graphics with parallel computing technology to create interactive distributed systems.